UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/644,357	08/20/2003	David Wendt	RSW920030138US1	6413

23307 7590 12/31/2007
SYNNESTVEDT & LECHNER, LLP
1101 MARKET STREET
26TH FLOOR
PHILADELPHIA, PA 19107-2950

EXAMINER	
HICKS, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2165	

MAIL DATE	DELIVERY MODE
12/31/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/644,357	WENDT, DAVID	
	Examiner	Art Unit	
	Michael J. Hicks	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 October 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *20 August 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-20 Pending.

Continued Examination Under 37 CFR 1.114

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 10/1/2007 has been entered.

Response to Arguments

3. Applicant's arguments filed 10/1/12007 have been fully considered but they are not persuasive.

As per Applicants arguments that Gong fails to disclose the limitation of returning the class file to a compiler to produce machine executable code, Examiner respectfully disagrees. Firstly, Examiner notes that the claims do not state any further processing is necessary, but only that the compiler executes said class data file to produce machine executable code. Examiner contends that an applet is machine executable code which is produced in response to the execution of the returned class file. Furthermore, in response to applicants comments that the JVM merely interprets the returned byte

codes and that no executable machine code is produced, Examiner points out that a compiler merely interprets byte codes in order to produce other data (e.g. an executable file). In the case of the JVM, the fact that the applet qualifies as executable code provides further evidence that the JVM is equatable to a compiler, as both, in essence, simply interpret information in order to transform the information.

In light of the above arguments, the rejection will be updated to reflect the amendments made to the claims, and maintained.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 8-20 rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

As per Claims 8-14, the claims fail to place the invention squarely within one statutory class of invention. On Page 9, Lines 14-19 of the instant specification, applicant has provided evidence that applicant intends the "computer program product" to include signals. As such, the claim is drawn to a form of energy. Energy is not one of the four categories of invention and therefore this claim(s) is/are not statutory. Energy is not a series of steps or acts and thus is not a process. Energy is not a

physical article or object and as such is not a machine or manufacture. Energy is not a combination of substances and therefor not a composition of matter.

As Per Claims 15-20, the claims lack the necessary physical articles or objects to constitute a machine or a manufacture within the meaning of 35 USC 101. They are clearly not a series of steps or acts to be a process nor are they a combination of chemical compounds to be a composition of matter. As such, they fail to fall within a statutory category. They are, at best, functional descriptive material *per se*. Note that the term system does not necessarily imply hardware, and the disclosure of Page 5, Lines 5-21 clearly indicate that the system is a compiler, e.g. software.

Descriptive material can be characterized as either "functional descriptive material" or "nonfunctional descriptive material." Both types of "descriptive material" are nonstatutory when claimed as descriptive material *per se*, 33 F.3d at 1360, 31 USPQ2d at 1759. When functional descriptive material is recorded on some computer-readable medium, it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized. Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994)

Merely claiming nonfunctional descriptive material, i.e., abstract ideas, stored on a computer-readable medium, in a computer, or on an electromagnetic carrier signal, does not make it statutory. See *Diehr*, 450 U.S. at 185-86, 209 USPQ at 8 (noting that the claims for an algorithm in *Benson* were unpatentable as abstract ideas because

"[t]he sole practical application of the algorithm was in connection with the programming of a general purpose computer.").

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

7. Claims 1-2, 4-9, 11-16, and 18-20 rejected under 35 U.S.C. 102(e) as being anticipated by Gong ("Secure Java Class Loading", IEEE Internet Computing, November/December 1998, Pgs. 56-61).

As per Claims 1, 8, and 15, Gong discloses a method, system and computer program product for compiling source code using a compiler having a classpath (i.e. *"Second, compilers and a bytecode verifier ensure that the Java virtual machine executes only legitimate Java code. The bytecode verifier, together with the Java virtual machine, guarantees language type safety at runtime. Moreover, a class loader defines a local name space, which helps to ensure that an untrusted applet cannot interfere with the running of other Java programs."* The preceding text excerpt clearly indicates compilers compile source code (e.g. byte code) which has class loaders (e.g. a class path).) (Page 57, Insert), comprising the steps of: 1) determining if a referenced class file is located in a workspace (i.e. *"Class loading has several unique characteristics. First, lazy loading means that classes are loaded on demand, on a just-in-time basis. Second, dynamic class loading maintains the type safety of the Java virtual machine by adding link-time checks, which replace certain runtime checks and are performed only once. Moreover, programmers can define their own class loaders that, for example, specify the remote location from which certain classes are loaded, or assign appropriate security attributes to them. Finally, programmers can use class loaders to provide separate name spaces for various software components. For example, a browser can load applets from different Web pages using separate class loaders, thus maintaining a degree of isolation between those applet classes. In fact, these applets can contain classes of the same name—the Java virtual machine treats these classes as distinct types."* The preceding text excerpt clearly indicates that to load a class, the classpath specified in the class loader is checked, and then the workspace indicated in the classpath is check to determine if the referenced class file is located there.) (Page 58, Column 2, Paragraph 3); 2) locating said class file in said workspace (i.e. *"Class loading has several unique characteristics. First, lazy loading means that classes are loaded on demand, on a just-in-time basis. Second, dynamic class loading maintains the type safety of the Java virtual machine by adding link-time checks, which replace certain runtime checks and are performed only once. Moreover, programmers can define their own class loaders that, for example, specify the remote location from which certain classes are loaded, or*

assign appropriate security attributes to them. Finally, programmers can use class loaders to provide separate name spaces for various software components. For example, a browser can load applets from different Web pages using separate class loaders, thus maintaining a degree of isolation between those applet classes. In fact, these applets can contain classes of the same name—the Java virtual machine treats these classes as distinct types." The preceding text excerpt clearly indicates that after it has been determined that the class file is present, it is located in the workspace.) (Page 58, Column 2, Paragraph 3); 3) accessing said class file (i.e. *"Class loading has several unique characteristics. First, lazy loading means that classes are loaded on demand, on a just-in-time basis. Second, dynamic class loading maintains the type safety of the Java virtual machine by adding link-time checks, which replace certain runtime checks and are performed only once. Moreover, programmers can define their own class loaders that, for example, specify the remote location from which certain classes are loaded, or assign appropriate security attributes to them. Finally, programmers can use class loaders to provide separate name spaces for various software components. For example, a browser can load applets from different Web pages using separate class loaders, thus maintaining a degree of isolation between those applet classes. In fact, these applets can contain classes of the same name—the Java virtual machine treats these classes as distinct types."* The preceding text excerpt clearly indicates that the class file is eventually loaded, which indicates that it is accessed and then returned to the compiler.) (Page 58, Column 2, Paragraph 3); and 4) returning said class file data to said compiler wherein said compiler executes said class data file to produce machine executable code (i.e. *"Class loading has several unique characteristics. First, lazy loading means that classes are loaded on demand, on a just-in-time basis. Second, dynamic class loading maintains the type safety of the Java virtual machine by adding link-time checks, which replace certain runtime checks and are performed only once. Moreover, programmers can define their own class loaders that, for example, specify the remote location from which certain classes are loaded, or assign appropriate security attributes to them. Finally, programmers can use class loaders to provide separate name spaces for various software components. For example, a browser can load applets from different Web pages using separate class loaders, thus*

maintaining a degree of isolation between those applet classes. In fact, these applets can contain classes of the same name—the Java virtual machine treats these classes as distinct types." The preceding text excerpt clearly indicates that after it has been determined that the class file is present, it is located in the workspace. Note that the class file is returned and executed in order to produce Java applet files (e.g. machine executable code.) (Page 58, Column 2, Paragraph 3).

As per Claims 2, 9, and 16, Gong discloses the step of locating said class file further comprises the steps of: identifying a location of a class using a workspace indicator in said classpath (i.e. *"Class loading has several unique characteristics. First, lazy loading means that classes are loaded on demand, on a just-in-time basis. Second, dynamic class loading maintains the type safety of the Java virtual machine by adding link-time checks, which replace certain runtime checks and are performed only once. Moreover, programmers can define their own class loaders that, for example, specify the remote location from which certain classes are loaded, or assign appropriate security attributes to them. Finally, programmers can use class loaders to provide separate name spaces for various software components. For example, a browser can load applets from different Web pages using separate class loaders, thus maintaining a degree of isolation between those applet classes. In fact, these applets can contain classes of the same name—the Java virtual machine treats these classes as distinct types."* The preceding text excerpt clearly indicates that a workspace indicator is located in the class loader (e.g. the programmer specified location for the workspace).) (Page 58, Column 2, Paragraph 3); and reading said class from said location (i.e. *"Class loading has several unique characteristics. First, lazy loading means that classes are loaded on demand, on a just-in-time basis. Second, dynamic class loading maintains the type safety of the Java virtual machine by adding link-time checks, which replace certain runtime checks and are performed only once. Moreover, programmers can define their own class loaders that, for example, specify the remote location from which certain classes are loaded, or assign appropriate security attributes to them. Finally, programmers can use class loaders*

to provide separate name spaces for various software components. For example, a browser can load applets from different Web pages using separate class loaders, thus maintaining a degree of isolation between those applet classes. In fact, these applets can contain classes of the same name—the Java virtual machine treats these classes as distinct types." The preceding text excerpt clearly indicates that the class is loaded from the given location.) (Page 58, Column 2, Paragraph 3).

As per Claims 4 and 11, Gong discloses the step of determining if a referenced class file is located in a workspace further comprises the steps of: reading an item from said classpath (i.e. *"To achieve this goal, JDK 1.2 distinguishes genuine system classes from all other classes by means of separate class paths. One is the system class path, for storing system classes. The other is the application class path, for storing all other classes. The Java virtual machine still loads classes on the system class path with the primordial class loader or a URLClassLoader and trusts them by default. A URLClassLoader usually loads classes on the application class path, and the Java virtual machine grants such classes the appropriate permissions according to the security policy."* The preceding text excerpt clearly indicates an item is read from the classpath to determine the location of the class.) (Page 61, Column 1, Paragraph 5); determining if said item references said file system or said workspace (i.e. *"To achieve this goal, JDK 1.2 distinguishes genuine system classes from all other classes by means of separate class paths. One is the system class path, for storing system classes. The other is the application class path, for storing all other classes. The Java virtual machine still loads classes on the system class path with the primordial class loader or a URLClassLoader and trusts them by default. A URLClassLoader usually loads classes on the application class path, and the Java virtual machine grants such classes the appropriate permissions according to the security policy."* The preceding text excerpt clearly indicates that the class loader determines whether the item is referencing a system, or workspace class by determining which classpath it was read from.) (Page 61, Column 1, Paragraph 5); searching a file system directory specified by said item if said item references said file

system (i.e. *"To achieve this goal, JDK 1.2 distinguishes genuine system classes from all other classes by means of separate class paths. One is the system class path, for storing system classes. The other is the application class path, for storing all other classes. The Java virtual machine still loads classes on the system class path with the primordial class loader or a URLClassLoader and trusts them by default. A URLClassLoader usually loads classes on the application class path, and the Java virtual machine grants such classes the appropriate permissions according to the security policy."* The preceding text excerpt clearly indicates that the item is located and loaded from either the system or the workspace as per the determination of where it is loacated.) (Page 61, Column 1, Paragraph 5); and searching said workspace if said item references said workspace (i.e. *"To achieve this goal, JDK 1.2 distinguishes genuine system classes from all other classes by means of separate class paths. One is the system class path, for storing system classes. The other is the application class path, for storing all other classes. The Java virtual machine still loads classes on the system class path with the primordial class loader or a URLClassLoader and trusts them by default. A URLClassLoader usually loads classes on the application class path, and the Java virtual machine grants such classes the appropriate permissions according to the security policy."* The preceding text excerpt clearly indicates that the item is located and loaded from either the system or the workspace as per the determination of where it is located.) (Page 61, Column 1, Paragraph 5).

As per Claims 5, 12, and 18, Gong discloses said class file data is contained in a database (i.e. *"An individual class representation is called a class file, even though it need not be stored in an actual file. For example, class files can be stored as records or commands in a database."* The preceding text excerpt clearly indicates that the class file may be contained in a database.) (Page 58, Column 2, Paragraph 4).

As per Claims 6, 13, and 19, Gong discloses said class file is contained within a .JAR file in said workspace (i.e. *"For example, on Unix systems, the class path can be set via the Shell environment variable CLASSPATH. Essentially, all classes or Java Archive files containing classes on the local file system must reside on this path to be discovered."* The preceding text excerpt clearly indicates that the class file may be contained in a Java Archive (e.g. .jar) file.) (Page 61, Column 1, Paragraph 3).

As per Claims 7, 14, and 20, Gong discloses said source code is Java (Note that the paper discusses Java classes and JDK, which indicates that the source code would be in Java.).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 3, 10, and 17 rejected under 35 U.S.C. 103(a) as being unpatentable over Gong in view of Bobbitt et al (U.S Pre Grant Publication Number 2003/0115218 and referred to hereinafter as Bobbitt).

As per Claims 3, 10, and 17, Gong fails to disclose said indicator comprises a signature string, a user ID, a project ID, and a workspace name.

Bobbitt discloses said indicator comprises a signature string, a user ID, a project ID, and a workspace name (i.e. *"The directory structure stored in Gossamer namespace parallels the virtual directory hierarchy, wherein the files contained (logically) in the virtual directories are replaced by file pointers having the same names as the original files...Accordingly, the respective file pointers to these files having the same namespace and located in the same subdirectory path ("/user/joe") relative to the /Namespace directory are stored in Gossamer namespace."* The preceding text excerpt clearly indicates that the classpath indicator, as disclosed above, may consist of a signature string (e.g. a pointer which identifies the file/class file in its virtual file system/workspace location) which consist of a user ID (e.g. joe in user/joe), a project ID (e.g. represented by user in /user) and a workspace name (e.g. represented by /Namespace).) (Page 5, Paragraph 0053).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Gong with the teachings of Bobbitt to include said indicator comprises a signature string, a user ID, a project ID, and a workspace name with the motivation of allowing access to files in a virtual file system (e.g. a workspace) by using a file pathname to identify the file and map it to a location which is accessible from outside the virtual file system (e.g. workspace) (Bobbitt, Page 1, Paragraph 8).

Points of Contact

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Hicks whose telephone number is (571) 272-2670. The examiner can normally be reached on Monday - Friday 8:30a - 5:00p.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey Gaffin can be reached on (571) 272-4146. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Michael J Hicks
Art Unit 2165
(571) 272-2670

CHRISTIAN CHACE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100